MolecuCrypt

Department of Information Technology
College of Computing and Information Sciences
Mariano Marcos State University

BENAVIDEZ, MARY CHRIS REYES
DALUZON, SHANTAL VENESS DE PADUA
VENTURA, JOHN CHRISTIAN GUBATAN

Date of Submission
May 20, 2025

Cultivating Minds, Transforming Futures

I. INTRODUCTION

In this age of digital communication, data security is a non-negotiable concern. MolecuCrypt is a locally developed encryption tool used to encrypt text into secure molecular form through ASCII conversion, bitwise shifting, and symbolic molecular representation. It has been coded in Python with Tkinter library and provides both encryption and decryption – with step-by-step manual decryption mode for teaching purposes.

The system not only safeguards information but also enables the user to learn basic cryptographic concepts by undergoing a symbol and visual change process. By integrating the logic of binary calculation with a chemistry approach, MolecuCrypt creates a new way of learning how encryption processes work internally.

II. ENCRYPTION LOGIC EXPLANATION

MolecuCrypt's encryption procedure transforms plaintext to a safe, encrypted molecular state in three general steps:

A. Text to Binary Conversion

- Each character is transferred into its ASCII 8-bit binary representation.

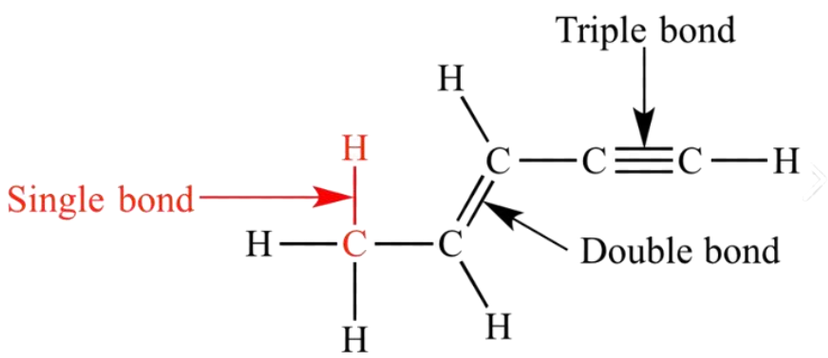
Example: “A” → 01000001

B. Circular Bit Shifting

- Binary chunks are split into bytes (8-bits) and shifted left circularly by the specified number of bits.

Example: Circular left shift by 2: **01**000001 → 00000**01**

C. Molecular Substitution



- The 0s and 1s are symbolically replaced to form a "molecular bond" representation:

0 → '–' (single bond)

1 → '=' (double bond)

In encryption, spaces represented as '≡' (triple bonds) only.

Example: 00000101 → "----==-="

The shifted binary (concatenated) is also converted back into ASCII to produce a secondary encrypted form.

III. DECRYPTION LOGIC EXPLANATION

A. Molecular to Binary Conversion

- '–' → 0
- '=' → 1

B. Reverse Circular Bit Shift

Binary is regrouped into 8-bit chunks and shifted right circularly using the same shift value.

Example: 00000101 → 01000001

C. Binary to Text Conversion

The resulting binary is then converted back to ASCII to recover the original plain text.

An optional manual decryption function shows these steps with intermediate outputs for better understanding.

IV. SECURITY CONSIDERATIONS AND JUSTIFICATIONS

Encryption technique employed by MolecuCrypt is customized obfuscation technique where plain text is converted via numerous irreversible-looking transformations, offering an added layer of protection appropriate for low-level applications like educational showcases or novelty coding. The reasons why the approach can be deemed secure for these kinds of application are as follows:

- **Multilayered Transformation:**
  - **Text to Binary:** The plain text is first changed into 8-bit binary, removing any readable patterns.
  - **Bit Shifting:** A user-defined value is used to circularly shift bits in each byte.

This step adds randomness and makes patterns harder to detect.

- **Molecular Encoding:** After shifting, the binary is converted into symbolic characters (‘0’ becomes ‘-‘ , ‘1’ becomes ‘=’, and byte separators become ‘≡’). This makes the binary data completely hidden and unrecognizable.

- **Non-Trivial Reversibility:** Without the correct bit shift value, decoding becomes much harder. A brute-force attack would require trying all 255 possible shifts for every byte.
- **Obfuscation from Human Readability:** The final encrypted output (which looks like molecular symbols) is completely different from the original text. To someone looking at it, it seems like nonsense unless they know the exact method and shift key.
- **Controlled Key Parameter:** The shift value (ranging from 1 to 255) acts like a key. Without knowing this, it’s very hard for someone to decrypt the message – especially since the symbolic format makes binary analysis nearly impossible.
- **Manual Decryption Transparency:** The built-in “Manual Decryption” feature shows each stage of the process, making the method great for learning purposes and helping students understand how the encryption works step-by-step.

V. CODE ANNONATIONS AND EXPLANATION

VI.

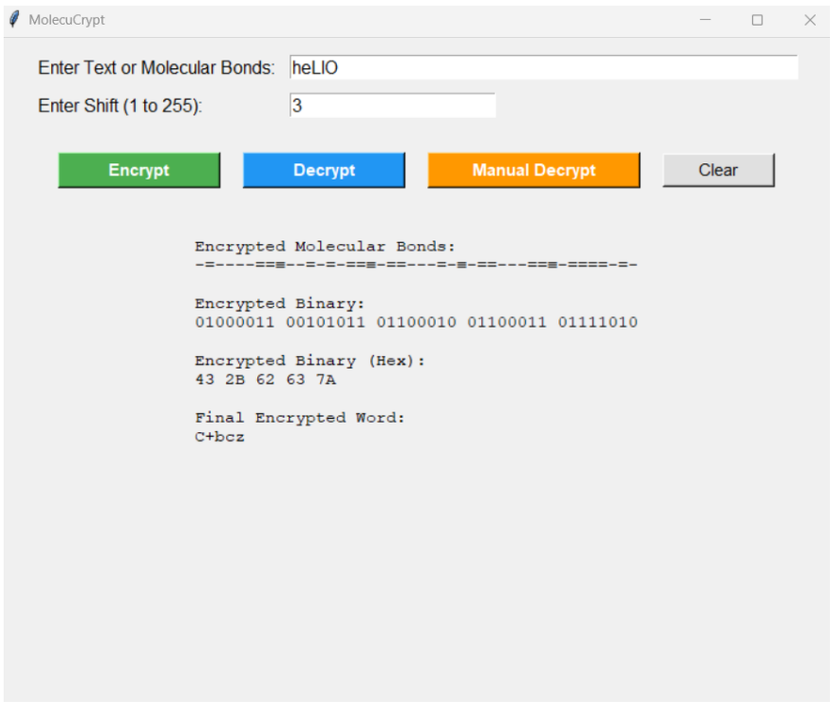
```
import tkinter as tk  # Import tkinter for GUI
from tkinter import messagebox  # Import messagebox for error dialogs

# --- Molecular/Bitshift Functions ---

def text_to_binary(text):
    # Convert each character in the text to its 8-bit binary
representation and join them into a string
    return ''.join(format(ord(char), '08b') for char in text)

def binary_to_text(binary):
    # Convert a binary string (without spaces) back to text, 8 bits at a
time
    return ''.join(chr(int(binary[i:i+8], 2)) for i in range(0,
len(binary), 8))

def shift_bits(binary, shift, direction="encrypt"):
    # Circularly shift each byte (8 bits) in the binary string
    shifted = ""
    for byte in binary.split():  # Split the binary string into bytes
        shift = shift % 8  # Ensure shift is within 0-7
        if direction == "encrypt":
            # For encryption, shift left
            shifted_byte = byte[shift:] + byte[:shift]
        else:
            # For decryption, shift right
            shifted_byte = byte[-shift:] + byte[:-shift]
```

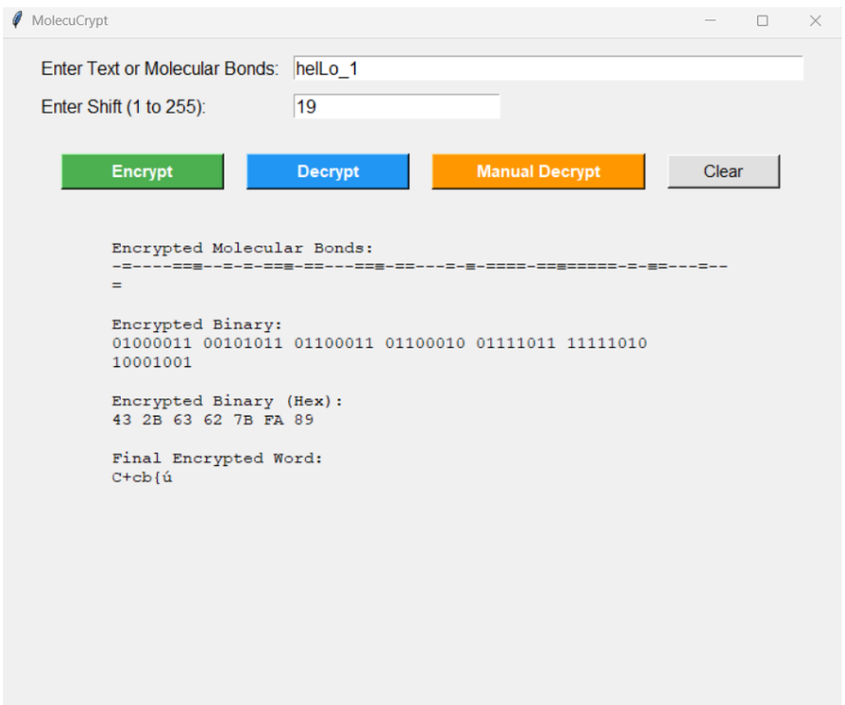
```
            shifted += shifted_byte + " "  # Add shifted byte to result
        return shifted.strip()  # Remove trailing space

    def binary_to_molecular(binary):
        # Encode binary to molecular format: 0 -> '-', 1 -> '=', space -> '≡'
        molecular = binary.replace('0', '-').replace('1', '=')
        return molecular.replace(" ", "≡")

    def molecular_to_binary(molecular):
        # Decode molecular format back to binary: '-' -> 0, '=' -> 1, '≡' ->
    space
        return molecular.replace('-', '0').replace('=', '1').replace('≡', ' ')

    def encrypt(text, shift=2):
        # Encrypt the input text using molecular encoding and bitwise shifting
        try:
            binary = text_to_binary(text)  # Convert text to binary
            # Split binary into bytes, shift, then encode
            shifted_binary = shift_bits(' '.join(binary[i:i+8] for i in
    range(0, len(binary), 8)), shift, "encrypt")
            molecular = binary_to_molecular(shifted_binary)  # Encode to
    molecular
            encrypted_word = binary_to_text(shifted_binary.replace(" ", ""))
    # Convert shifted binary to text
            return molecular, shifted_binary, encrypted_word  # Return all
    representations
        except Exception as e:
            return f"Error during encryption: {e}", "", ""

    def decrypt(molecular, shift=2):
        # Decrypt the molecular string using the provided shift value
        try:
            binary = molecular_to_binary(molecular)  # Decode molecular to
    binary
            shifted_binary = shift_bits(binary, shift, "decrypt")  # Reverse
    shift
            return binary_to_text(shifted_binary.replace(" ", ""))  # Convert
    to text
        except Exception as e:
            return f"Error during decryption: {e}"

    def manual_decrypt(molecular, shift=2):
        # Show step-by-step decryption for educational purposes
        try:
            binary = molecular_to_binary(molecular)  # Step 1: molecular to
    binary
            shifted_binary = shift_bits(binary, shift, "decrypt")  # Step 2:
    reverse shift
            text = binary_to_text(shifted_binary.replace(" ", ""))  # Step 3:
    binary to text
            return f"Step 1: Molecular to Binary: {binary}\n" \
                   f"Step 2: Reverse Bit Shift: {shifted_binary}\n" \
                   f"Step 3: Binary to Text: {text}"
        except Exception as e:
            return f"Error during manual decryption: {e}"

    def handle_clear():
        # Clear all input and output fields
        entry_text.delete(0, tk.END)
        entry_shift.delete(0, tk.END)
        result_label.config(text="Result will be displayed here.")


    # --- GUI Functions ---

    def handle_encrypt():
        # Handle the Encrypt button click event
```

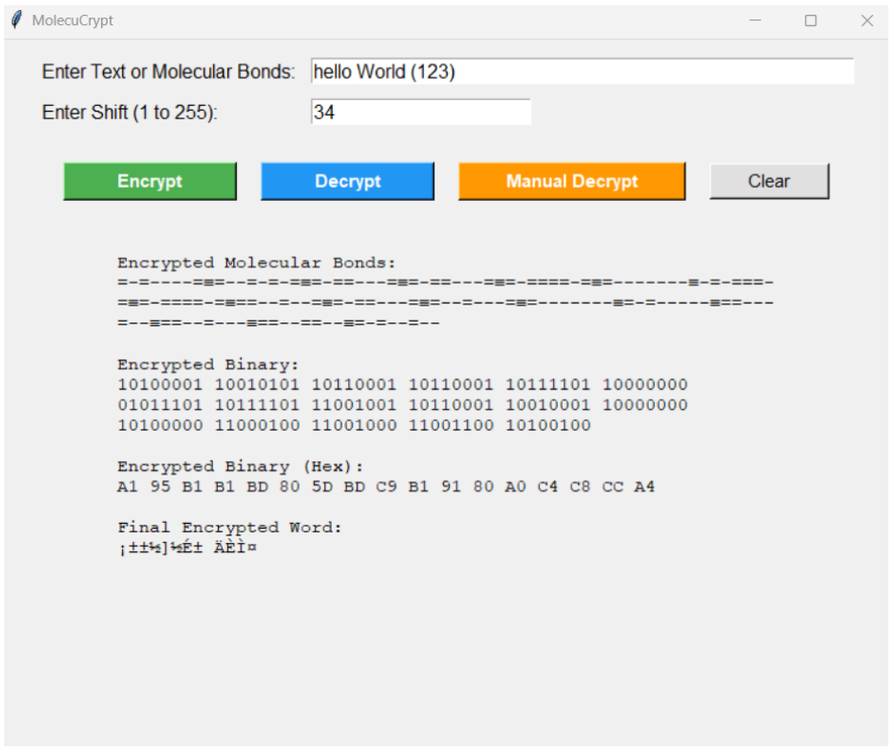
```
    text = entry_text.get()  # Get text from input field
    shift = entry_shift.get()  # Get shift value from input field
    if not text:
        messagebox.showerror("Error", "Please enter text to encrypt.")  #
Show error if text is empty
        return
    if not shift.isdigit() or int(shift) <= 0 or int(shift) > 255:
        messagebox.showerror("Error", "Shift must be a positive integer
between 1 and 255.")  # Validate shift
        return
    molecular, shifted_binary, encrypted_word = encrypt(text, int(shift))
# Encrypt the text
    # Convert each 8 bits (byte) of shifted_binary to hex, separated by
spaces
    hex_bytes = []
    for byte in shifted_binary.split():
        hex_byte = hex(int(byte, 2))[2:].upper()  # Convert byte to hex
        if len(hex_byte) == 1:
            hex_byte = "0" + hex_byte  # Pad single digit hex
        hex_bytes.append(hex_byte)
    hex_output = " ".join(hex_bytes)  # Join hex bytes
    # Display all results in the result label
    result_label.config(text=f"Encrypted Molecular
Bonds:\n{molecular}\n\n"
                            f"Encrypted Binary:\n{shifted_binary}\n\n"
                            f"Encrypted Binary (Hex):\n{hex_output}\n\n"
                            f"Final Encrypted Word:\n{encrypted_word}")

def handle_decrypt():
    # Handle the Decrypt button click event
    molecular = entry_text.get()  # Get molecular string from input field
    shift = entry_shift.get()  # Get shift value from input field
    if not molecular:
        messagebox.showerror("Error", "Please enter molecular bonds to
decrypt.")  # Show error if empty
        return
    if not shift.isdigit() or int(shift) <= 0 or int(shift) > 255:
        messagebox.showerror("Error", "Shift must be a positive integer
between 1 and 255.")  # Validate shift
        return
    decrypted = decrypt(molecular, int(shift))  # Decrypt the molecular
string
    result_label.config(text=f"Decrypted Text:\n{decrypted}")  # Display
result

def handle_manual_decrypt():
    # Handle the Manual Decrypt button click event (shows step-by-step)
    molecular = entry_text.get()  # Get molecular string from input field
    shift = entry_shift.get()  # Get shift value from input field
    if not molecular:
        messagebox.showerror("Error", "Please enter molecular bonds to
decrypt.")  # Show error if empty
        return
    if not shift.isdigit() or int(shift) <= 0 or int(shift) > 255:
        messagebox.showerror("Error", "Shift must be a positive integer
between 1 and 255.")  # Validate shift
        return
    steps = manual_decrypt(molecular, int(shift))  # Get step-by-step
decryption
    result_label.config(text=f"Manual Decryption Steps:\n\n{steps}")  #
Display steps

# --- GUI Setup ---

root = tk.Tk()  # Create main window
root.title("MolecuCrypt")  # Set window title
root.geometry("750x600")  # Set window size
```

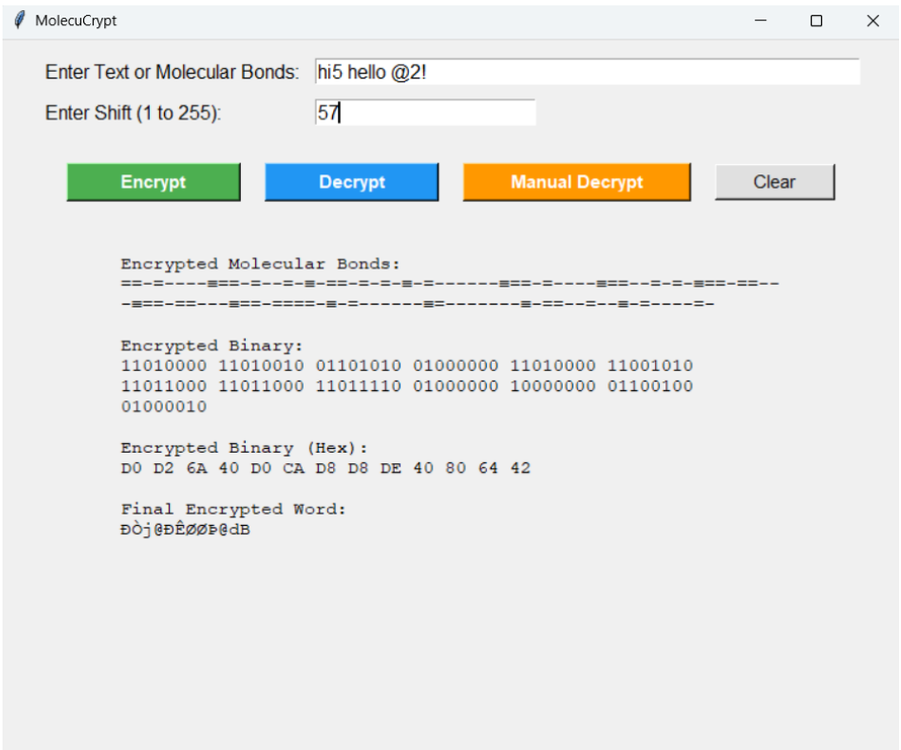
```
input_frame = tk.Frame(root)  # Create input frame
input_frame.pack(pady=10)  # Add padding

# Label and entry for text or molecular bonds
tk.Label(input_frame, text="Enter Text or Molecular Bonds:",
font=("Helvetica", 12)).grid(row=0, column=0, sticky="w", padx=5, pady=5)
entry_text = tk.Entry(input_frame, width=50, font=("Helvetica", 12))
entry_text.grid(row=0, column=1, padx=5, pady=5)

# Label and entry for shift value
tk.Label(input_frame, text="Enter Shift (1 to 255):", font=("Helvetica",
12)).grid(row=1, column=0, sticky="w", padx=5, pady=5)
entry_shift = tk.Entry(input_frame, width=20, font=("Helvetica", 12))
entry_shift.grid(row=1, column=1, padx=5, pady=5, sticky="w")

button_frame = tk.Frame(root)  # Create button frame
button_frame.pack(pady=15)

# Encrypt button
tk.Button(button_frame, text="Encrypt", command=handle_encrypt,
          width=15, font=("Helvetica", 11, "bold"), bg="#4CAF50",
fg="white").grid(row=0, column=0, padx=10)

# Decrypt button
tk.Button(button_frame, text="Decrypt", command=handle_decrypt,
          width=15, font=("Helvetica", 11, "bold"), bg="#2196F3",
fg="white").grid(row=0, column=1, padx=10)

# Manual Decrypt button
tk.Button(button_frame, text="Manual Decrypt",
command=handle_manual_decrypt,
          width=20, font=("Helvetica", 11, "bold"), bg="#FF9800",
fg="white").grid(row=0, column=2, padx=10)

# Manual Decrypt button
tk.Button(button_frame, text="Manual Decrypt",
command=handle_manual_decrypt,
          width=20, font=("Helvetica", 11, "bold"), bg="#FF9800",
fg="white").grid(row=0, column=2, padx=10)

# Clear button
tk.Button(button_frame, text="Clear", command=handle_clear,
          width=10, font=("Helvetica", 11), bg="#E0E0E0",
fg="black").grid(row=0, column=3, padx=10)

result_frame = tk.Frame(root)  # Create result frame
result_frame.pack(pady=15, fill="both", expand=True)

# Label to display results
result_label = tk.Label(result_frame, text="Result will be displayed
here.", wraplength=550, justify="left", font=("Courier New", 11))
result_label.pack(padx=10, pady=10)

root.mainloop()  # Start the GUI event loop
```

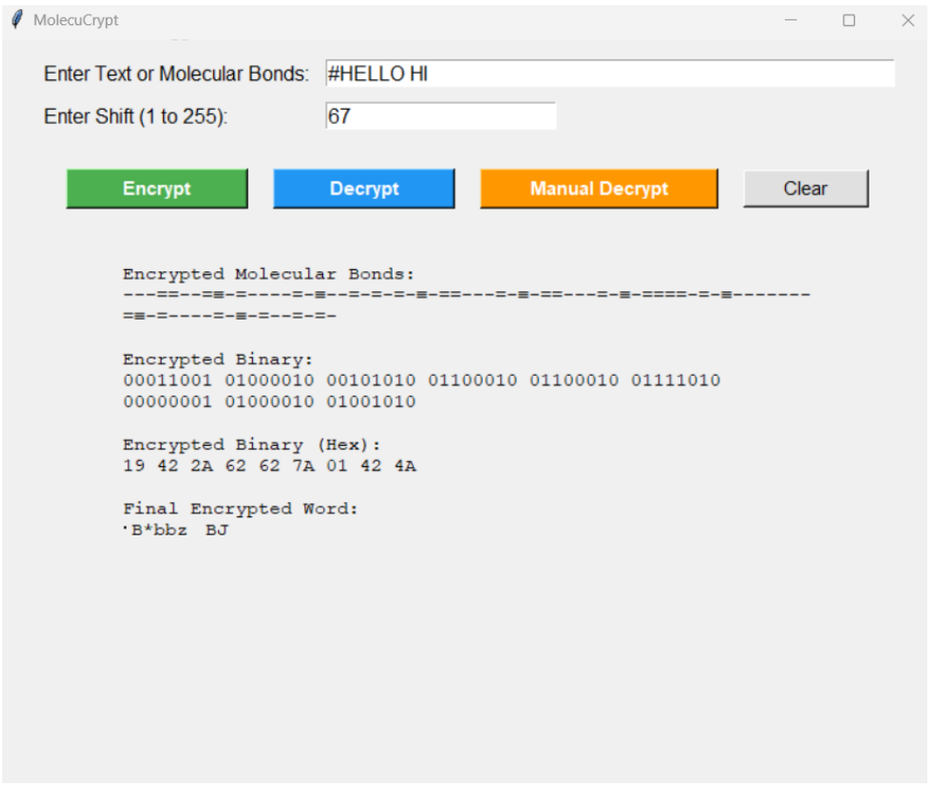
SAMPLE ENCRYPTION



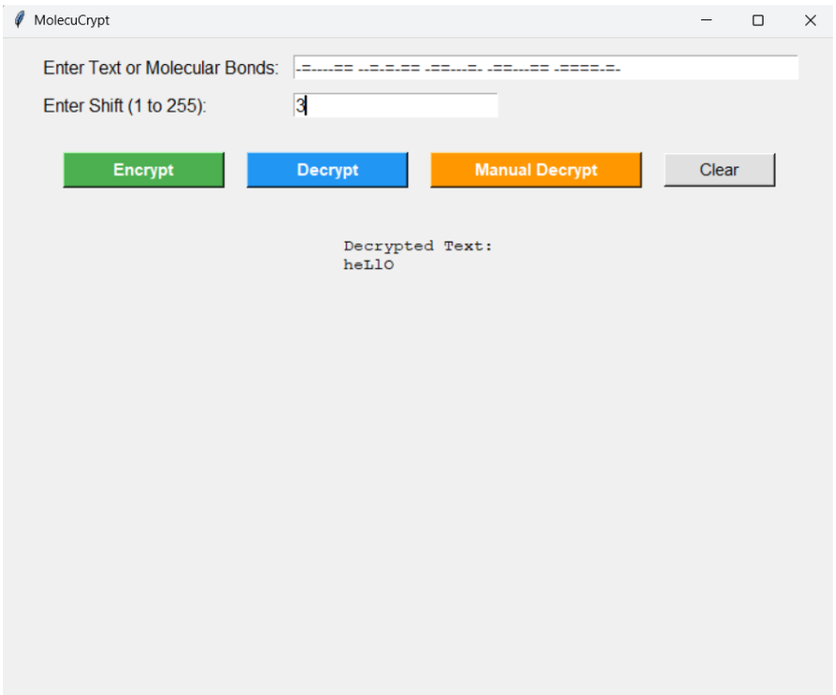


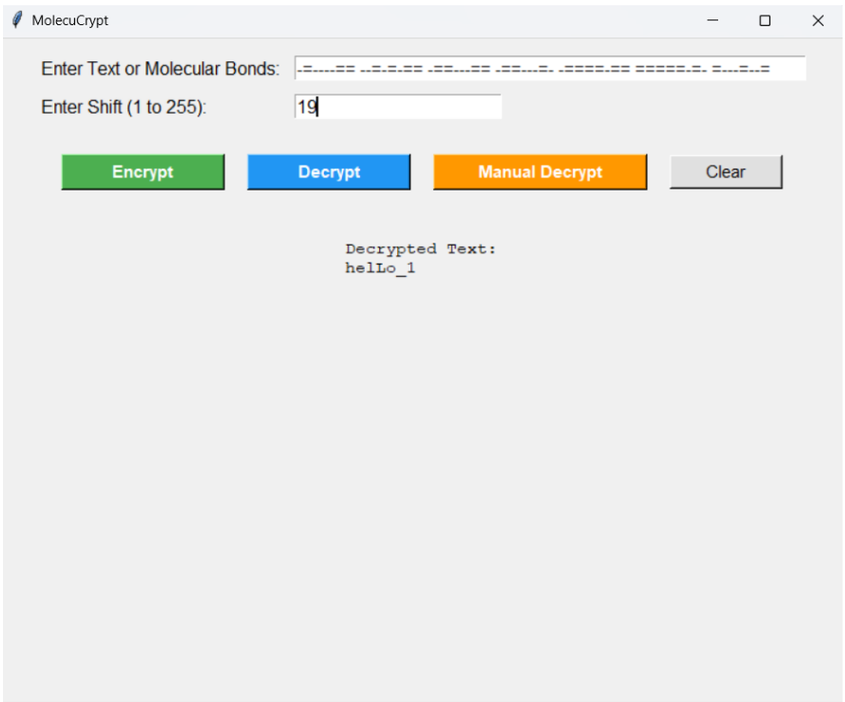

MolecuCrypt
Enter Text or Molecular Bonds:
hello World (123)
Enter Shift (1 to 255):
34
Encrypt
Decrypt
Manual Decrypt
Clear
Encrypted Molecular Bonds:
Encrypted Binary:
10100001 10010101 10110001 10110001 10111101 10000000
01011101 10111101 11001001 10110001 10010001 10000000
10100000 11000100 11001000 11001100 10100100
Encrypted Binary (Hex):
A1 95 B1 B1 BD 80 5D BD C9 B1 91 80 A0 C4 C8 CC A4
Final Encrypted Word:

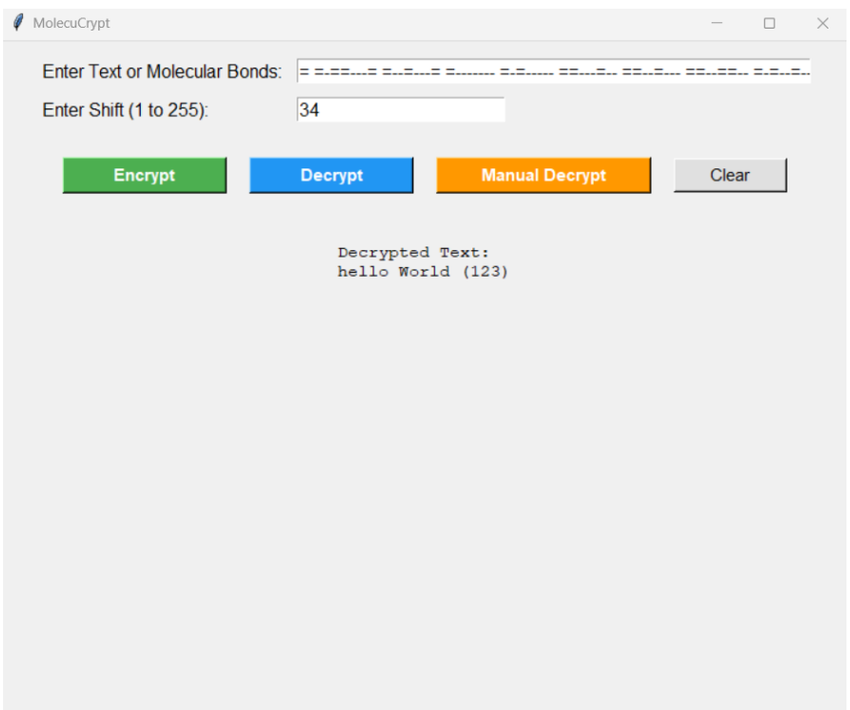

MolecuCrypt
Enter Text or Molecular Bonds:
hi5 hello @2!
Enter Shift (1 to 255):
57
Encrypt
Decrypt
Manual Decrypt
Clear
Encrypted Molecular Bonds:
Encrypted Binary:
11010000 11010010 01101010 01000000 11010000 11001010
11011000 11011000 11011110 01000000 10000000 01100100
01000010
Encrypted Binary (Hex):
D0 D2 6A 40 D0 CA D8 D8 DE 40 80 64 42
Final Encrypted Word:

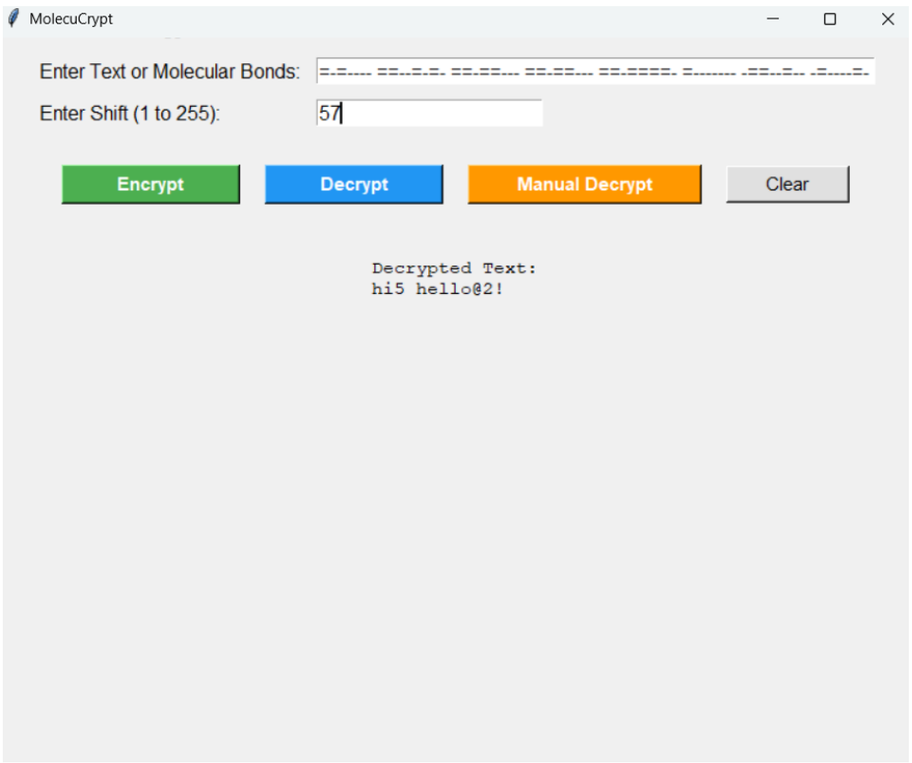


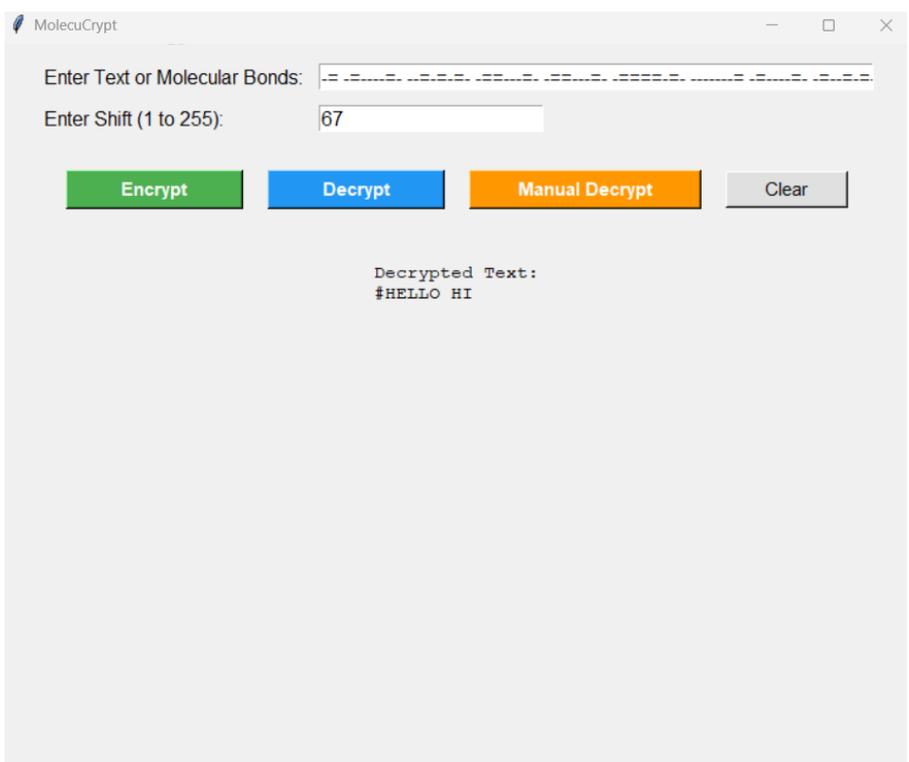
SAMPLE DECRYPTION

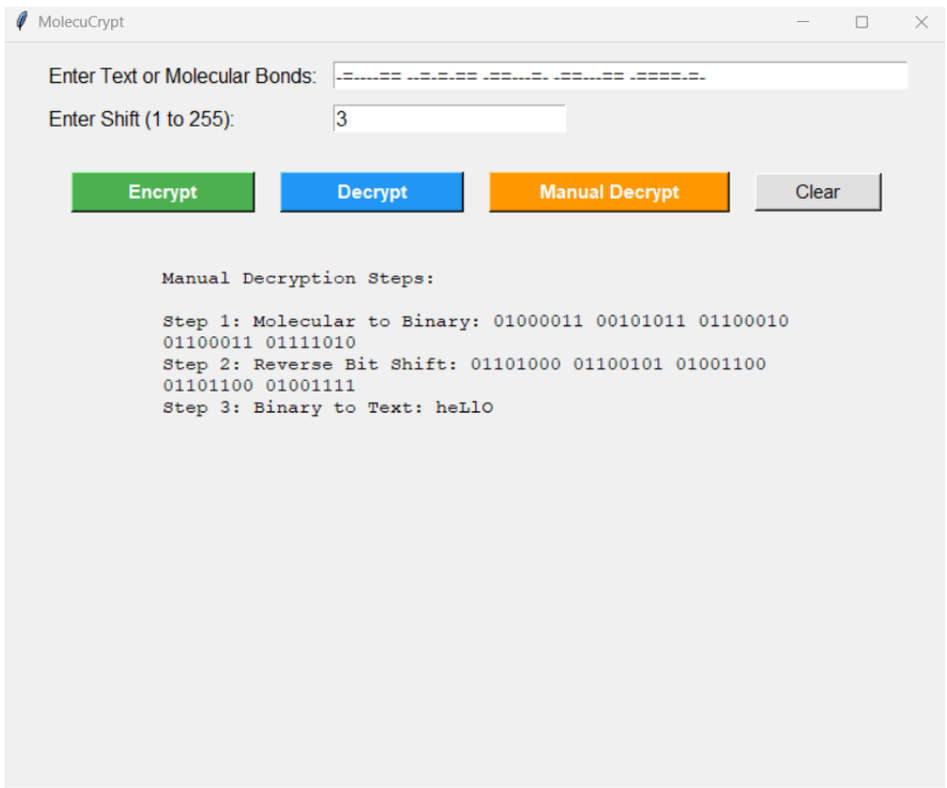


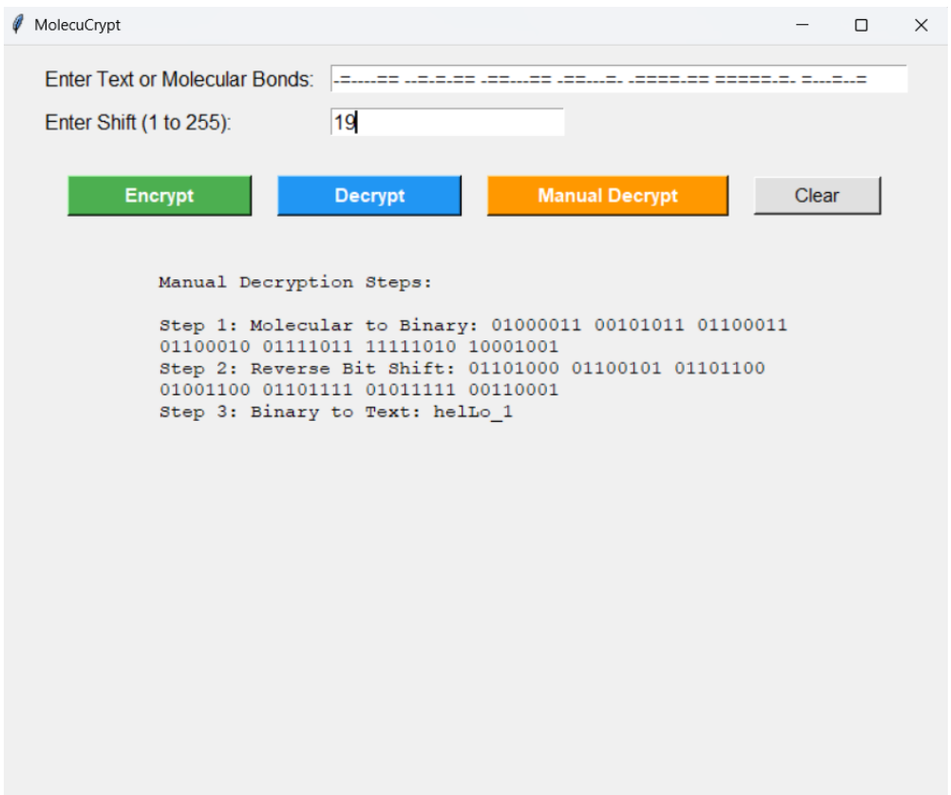

MolecuCrypt
Enter Text or Molecular Bonds:
Enter Shift (1 to 255):
19
Encrypt
Decrypt
Manual Decrypt
Clear
Decrypted Text:
helLo_1

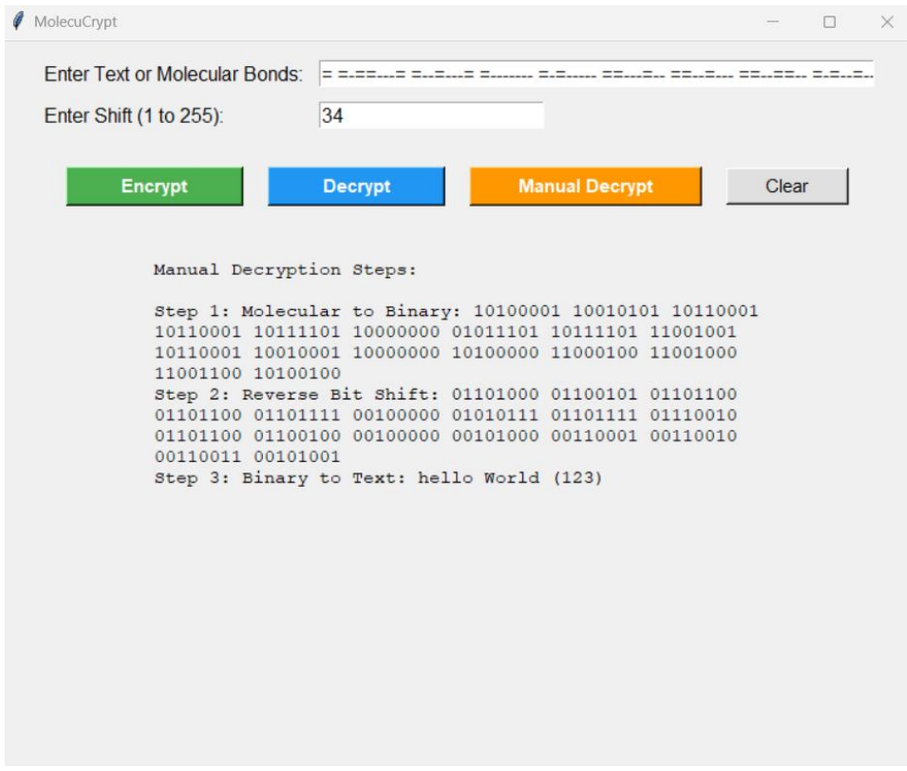

MolecuCrypt
Enter Text or Molecular Bonds:
Enter Shift (1 to 255):
34
Encrypt
Decrypt
Manual Decrypt
Clear
Decrypted Text:
hello World (123)

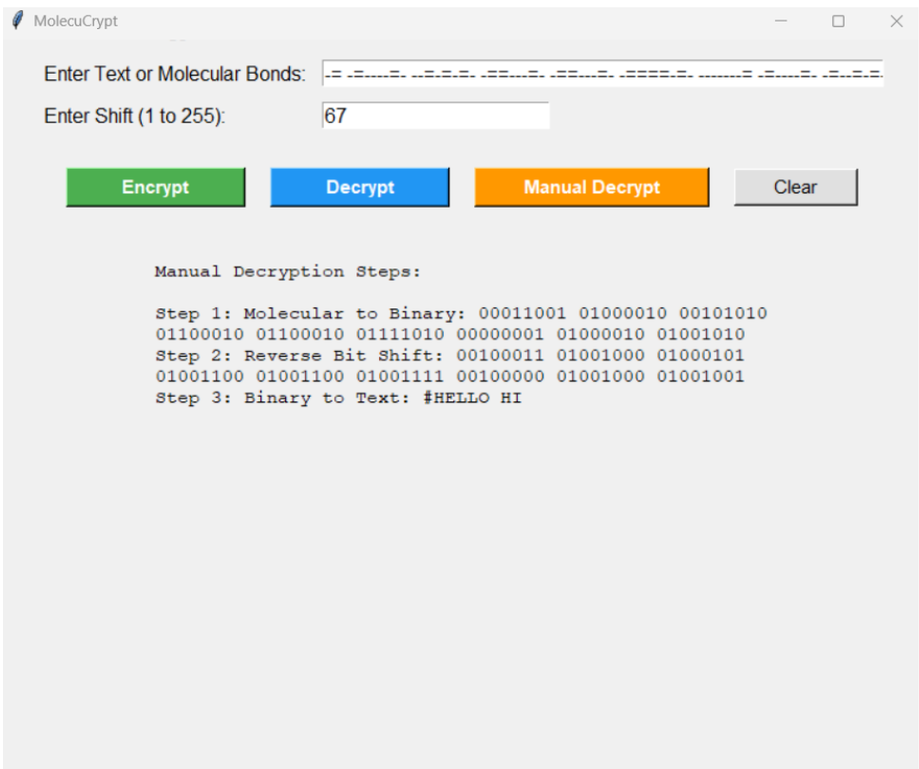

MolecuCrypt
Enter Text or Molecular Bonds:
Enter Shift (1 to 255):
57
Encrypt
Decrypt
Manual Decrypt
Clear
Decrypted Text:
hi5 hello@2!


MolecuCrypt
Enter Text or Molecular Bonds:
Enter Shift (1 to 255):
67
Encrypt
Decrypt
Manual Decrypt
Clear
Decrypted Text:
#HELLO HI


MolecuCrypt
Enter Text or Molecular Bonds:
-=-----== --=-=-== -==---=- -==---=- -=====-=-
Enter Shift (1 to 255):
3
Encrypt
Decrypt
Manual Decrypt
Clear
Manual Decryption Steps:
Step 1: Molecular to Binary: 01000011 00101011 01100010
01100011 01111010
Step 2: Reverse Bit Shift: 01101000 01100101 01001100
01101100 01001111
Step 3: Binary to Text: heLlO


MolecuCrypt
Enter Text or Molecular Bonds:
-=-----== --=-=-== -==---=- -==---=- -=====-== =====-=- =---=--=
Enter Shift (1 to 255):
19
Encrypt
Decrypt
Manual Decrypt
Clear
Manual Decryption Steps:
Step 1: Molecular to Binary: 01000011 00101011 01100011
01100010 01111011 11111010 10001001
Step 2: Reverse Bit Shift: 01101000 01100101 01101100
01001100 01101111 01011111 00110001
Step 3: Binary to Text: helLo_1


MolecuCrypt
Enter Text or Molecular Bonds:
Enter Shift (1 to 255):
34
Encrypt
Decrypt
Manual Decrypt
Clear
Manual Decryption Steps:
Step 1: Molecular to Binary: 10100001 10010101 10110001
10110001 10111101 10000000 01011101 10111101 11001001
10110001 10010001 10000000 10100000 11000100 11001000
11001100 10100100
Step 2: Reverse Bit Shift: 01101000 01100101 01101100
01101100 01101111 00100000 01010111 01101111 01110010
01101100 01100100 00100000 00101000 00110001 00110010
00110011 00101001
Step 3: Binary to Text: hello World (123)


MolecuCrypt
Enter Text or Molecular Bonds:
Enter Shift (1 to 255):
67
Encrypt
Decrypt
Manual Decrypt
Clear
Manual Decryption Steps:
Step 1: Molecular to Binary: 00011001 01000010 00101010
01100010 01100010 01111010 00000001 01000010 01001010
Step 2: Reverse Bit Shift: 00100011 01001000 01000101
01001100 01001100 01001111 00100000 01001000 01001001
Step 3: Binary to Text: #HELLO HI